

## البرامج الفرعية

عند التعامل مع البرامج الفرعية بلغة C++ هناك برامج فرعية تعيد قيمة وبرامج فرعية لا تعيد قيمة والتي تسمى بالإجراء `procedure` بلغة ترينبا. كان لغت C++ تتعامل مع الدوال فقط لذلك نقول أنه لدينا دالة تعيد قيمة ويتم ذلك باستخدام الأمر `return`.

دالة (القيمة المائدة) `return`

- هناك دوال لا تعيد قيمة ويتم ذلك باستخدام الكلمة المجردة `Void`.  
- `Void` هو أحد أنواع المتغيرات، مقدمة بلغة C++ وتستخدم للأشياء التي تستخدم مع الدوال التي لا تعيد قيمة وتستخدم ارتباطاً مع المؤشرات.

- تستخدم الدوال من أجل اختصار البرامج توفير لذاكرة سرعة التنفيذ وسهولة كشف الأخطاء.

- عند تعريف الدوال بلغة C++ عادة يتم تعريف الدوال في مقدمة البرنامج أما شرح الدوال يتم في نهاية البرنامج الرئيسي مع أنه يمكن شرح الدوال في المقدمة.

- من أجل تعريف البرامج الفرعية أو الدوال يتم ذلك كما يلي:

[1] - حدد نوع المتغيرات للدالة وعند عدم تحديد نوع المتغيرات تكون الدالة من نوع `int` وإذا كانت الدالة لا تعيد قيمة فحدد نوع المتغيرات `Void`.

[2] - رتب اسم اختياري للدالة إلى نوع المتغيرات.

[3] - رتب اسم الدالة قائمة من الشرطة أو أشرطة إذا وجدت محصورات قوسين ( ) وعند عدم وجود الشرطة تترك الأقواس فارغة.

[4] - نكتب الأمر الخاص بالدالة محصورة بين قوسين ابتدائية ونهاية على شكل `return` الأمر.

دالة القيمة `return`

هنا ضاع الأمر في الدالة إذا كان المطلوب من الدالة إعادة قيمة أي:



```

    {
    ...
    return (القيمة);
    }

```

وصفيتها:

(الوسيط) اسم الدالة ... نوع المتغيرات

```

{
    ...
    }

```

مثال توضيحي:

البارامترات (الوسيط)

```

int MM (int a, char c)
{
    int m1, m2;
    char cl;
    ...
    return (القيمة);
}

```

متغيرات

```

return (القيمة);
}

```

مثال:

```

void SS (int k)
{
    char c;
    ...
}

```

```

}

```

على أنه يجب استكمال البرنامج بالشكل التالي:

```

#include <iostream.h>
int MM (int a, char c);

```

void SS(int k);

void main ( )

{

long k1, k2;

استدعاء للدوال

}

int MM(int a, char c)

{

int m1, m2;

char c1;

return ( القيمة );

}

void SS(int k)

{

char c;

}

اثنان تعريف للدالة في  
مقدمة البرنامج تقع  
فاصلية منقوطة وعند  
نحو الدالة لا تقع  
فاصلية منقوطة

بعد نهاية  
البرنامج تبدأ  
شرح  
الدوال

ملاحظة:

(1) - يجب ان هذه تطابق وتوافقة بين نوعي المتغيرات للدالة ونوع القيمة  
بواسطة return

(2) - لا يجوز تعريف دالة باخرى وانما تعرف الدوال بشكل مستقل  
داخل دالة

(3) - عند التعامل مع الدوال الرمزية (char) يمكن التعامل مع الرقم او رقم  
الترتيب لمنه بدل ارسلي كود



## البارامترات الفعلية والبارامترات الشكلية

عند تعريف الدالة حدد نوع المتغيرات ثم نضف اسم الدالة وبيئ قوسين نذكر  
الوسائط أو البارامترات وهذه البارامترات هي بارامترات شكلية.  
لما عند استدعاء الدالة من داخل البرنامج الرئيسي يتم ذلك بذكر اسم الدالة  
تليها قائمة من الوسائط أو البارامترات وهذه البارامترات هي بارامترات فعلية.

• مثال :

و (int MM (char c, long a) بارامترات شكلية

و (S = MM ('!', 8) بارامترات فعلية

- يجب أن يحد تطابق وتوافق بين البارامترات الفعلية والبارامترات  
الشكلية من حيث العدد والنوع.

• مثال :

و (void MM (int a1, int a2, int a3;  
MM (2, 3, 6);

نستطيع المتغيرات المحلية والمتغيرات العامة  
المتغيرات المحلية : هي متغيرات تعرف للدالة ولتتم عمل هذه المتغيرات على

الدالة فقط أي لا يجوز استخدامها في دالة أخرى ولا يجوز استخدامها في  
البرنامج الرئيسي.

• تمرين :

باستخدام معنوم الدالة اكتب برنامج يحسب مجموع عددين ويكتب  
عدد المجموع.

```
#include <iostream.h>
```

```
int sum (int a, int b);
```

```
int cb (int k);
```

```
Void main ()
```



```

{
    int x, y, z, S1, S2;
    cout << "ln x = "; cin >> x;
    cout << "ln y = "; cin >> y;
    cout << "ln z = "; cin >> z;
    S1 = sum(x, y);
    S2 = cb(z);
    cout << "ln S1 = " << S1;
    cout << "ln S2 = " << S2;
}

int sum(int a, int b)
{
    int c;
    c = a + b;
    return (c);
}

int cb(int k)
{
    int t;
    t = k * k * k;
    return (t);
}

```

مکرمه  
 با احترام و تقدیر، لایحه اکتب برتاج به لایحه لغز لغزیت  
 لک طیاره BerLand C++ ۶ مرات:

```

#include <iostream.h>
int min(int a, int b);
void show();
    طیاره و لایحه

```

```

void main()
{
    int x, y, m;
    cout << "In x = "; cin >> x;
    cout << "In y = "; cin >> y;
    m = min(x, y);
    Show(m);
    cout << "In m = " << m;
}

```

```

int main(int a, int b)
{
    int k;
    if (a < b)
        k = a;
    else
        k = b;
    return(k);
}

```

يقول أن تكون  
الوسطاء غير  
المتغيرات

```

void Show()
{

```

```

    int i;
    for (i = 1; i <= 5; ++i)
        cout << "In Borland C++";
}

```

ملاحظة: يمكن للوسيط أن يكون دالة أو متغير.

بالإضافة إلى ذلك، يمكن استخدام الدالة والوسيط كمتغير أو كدالة.

```

#include <iostream.h>
#define n 5

```



## محاضرات الدفتر

المحاضرة :

المادة :

السنة :

القسم :

```
int sum (int a[5]);
```

```
void main ( )
```

```
{
```

```
int b[5], i, s = 0;
```

```
for (i = 0; i < 5; ++i)
```

```
cin >> b[i];
```

```
s = sum (b);
```

```
cout << "ln s = " << s;
```

```
}
```

```
int sum (int a[5])
```

```
{
```

```
int k, j;
```

عداد في الالة ليحسب مجموع المصفوفة.

```
k = 0;
```

```
for (j = 0; j < 5; ++j)
```

```
k = k + a[j];
```

```
return (k);
```

```
}
```

عند التعامل مع الالة

وإنما يتم اوصاف

المصفوفة في البرنامج

الرئيس أو ما جمع وفهرس

مصفوفتين يتم في خرج

الالة

إذا كان لو بسيط

عبارة عن مصفوفة

تذكر اسم

المصفوفة

من دون دليل

تربيع وطريقة:

باستخدام مفهوم الالة والمصفوفة اكتب برنامجاً<sup>(١)</sup> لحساب القيمة الصغرى لمصفوفة.

(٢) - في مصفوفة فربعد اجمع

(٣) - اجمع مصفوفتين ذات بعد واحد.

نسب دوال الإعادة الذاتية:

اتناد كتابة البرنامج يمكن في الواقع استخدام الالة لنفسها أكثر من مرة

وتستخدم عادة دوال الإعادة الذاتية في العديد من التطبيقات

عند استخدام دوال الإعادة ذاتية نضع مقترح التوقف.



5!

مثال: برنامہ بنائیں!

```

#include <iostream.h>
int Fact (int x);
void main ()
{
    int n, f;
    cin >> n;
    f = Fact (n);
    cout << "In F = " << f;
}
int Fact (int x)
{
    if (x <= 1)
        return (1);
    else
        return (x * Fact (x-1));
}

```

طریقہ، نتیجہ

```

int Fact (int 5)
5 = Fact (5);
= 5 * Fact (4)
= 5 * 4 * Fact (3)
= 5 * 4 * 3 * Fact (2)
= 5 * 4 * 3 * 2 * Fact (1)
= 120

```

ب

```

if (b == 0)
    return (1);
else
    a * pow (a, b-1)

```

نتیجہ

۴۷